Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Anže Veršnik

Nadzorna aplikacija za dolgoročno spremljanje bolnikov z nevrodegenerativnimi boleznimi

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Aleksander Sadikov
Somentor: doc. dr. Jure Žabkar

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Kandidat naj zasnuje in implementira spletno aplikacijo za dolgoročno spremljanje bolnikov z nevrodegenerativnimi boleznimi. Aplikacija naj zagotavlja varnost podatkov, avtentikacijo uporabnikov, možnost obdelave podatkov in podatkovno bazo v katere bomo podatke hranili podatke o pacientih in opravljenih meritvah. Narejena naj bo v obliki pregledne plošče, ki omogoča hiter in lahek pregled podatkov uporabniku v grafični ali tabelarni obliki. Aplikacija naj vključuje tudi administrativni del, kjer se dodajajo novi uporabniki aplikacije in beleženje akcij trenutnih uporabnikov.

Rad bi se zahvalil svojemu mentorju doc. dr. Aleksandru Sadikovu in somentorju doc. dr. Juretu Žabkarju za podporo in pomoč pri izdelavi diplomske naloge. Zahvalil bi se tudi zdravniku dr. Dejanu Georgievu za pomoč pri izdelavi diplomske naloge z medicinske plati. Posebej bi se rad zahvalil družini in prijateljem, ki so mi pomagali in me podpirali pri izdelavi diplomske naloge, kot tudi med študijem samim.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
ASP	active server pages	
RAD	rapid application development	hitri razvoj programov
HTML	hyper text markup language	
XML	extensible markup language	
PHP	PHP: Hypertext Preprocessor	
JSP	javaserver pages	
URL	Uniform Resource Locator	enolični krajevnik vira
HTTP	hypertext transfer protocol	protokol za prenos hiperteksta
CRUD	create, read, update and delete	ustvarjanje, branje, posodabljanje in brisanje
UTF	unicode transformation format	
TLS	transport layer security	
ID	identifier	identifikator
SSL	secure sockets layer	

Povzetek

Naslov: Nadzorna aplikacija za dolgoročno spremljanje bolnikov z nevrodegenerativnimi boleznimi

Avtor: Anže Veršnik

Zdravniki se dandanes vedno bolj zanašajo na pomoč informacijskih sistemov. Ti so lahko v pomoč zdravniku pri diagnozi bolezni, pridobivanju dodatnih informacij o bolezni, pridobivanju podatkov o pacientih in njihovem dosedanjem zdravljenju, načinih zdravljenja,... Omogočajo preglednost, hitrejši dostop in urejenost informacij ter hitrejše izmenjevanje informacij med zdravniki. Cilj te diplomske naloge je bil razvoj nadzorne plošče, ki bi omogočala zdravniku enostaven in uporaben pregled ter analizo podatkov o bolnikih s parkinsonovo boleznijo. S tem bi lahko hitreje in učinkoviteje zaznali in sledili spremembam stanja bolnika v določenem časovnem obdobju. Aplikacija to naredi s prikazom različnih statistik o bolnikovem stanju, tako grafično, kot tudi v tabeli. Zdravniku omogoča primerjanje bolezenskih stanj večih bolnikov ter ugotavljanja korelacij med posameznimi stanji. V diplomski nalogi je predstavljen razvoj te aplikacije.

Ključne besede: spletna aplikacija, Tomcat, Java, analiza podatkov, nadzorna plošča.

Abstract

Title: A dashboard application for long-term monitoring of patients with neurodegenerative diseases

Author: Anže Veršnik

Nowadays, doctors rely more and more on the help of information systems as part of their work. These systems can be of great help to the doctor in various scenarios. For example, they can aid him with the diagnosis of a disease, the obtaining of information concerning a disease and information about patients and their past medical treatment, as well as, the methods of treatment. All in all, they enable efficiency, faster access, organised information and faster exchange of information between doctors. The goal of this thesis was the development of a dashboard that would enable doctors a simple and practical overlook and the analysis of information concerning patients with the Parkinson's disease. With this kind of application, we would be able to recognise and track the changes of the state of a patient in a certain period in an easier and more efficient way. The application enables us to do this through different statistics about the state of the patient, both in graphs and charts. It also lets the doctor to compare disease states of multiple patients and to establish the correlation between individual states. The development of this application is discussed in this thesis.

Keywords: online application, Tomcat, Java, data analysis, dashboard.

Poglavje 1

Uvod

1.1 Motivacija za aplikacijo

Osebno me je vedno zanimalo delo na multidisciplinarnih področjih. To je bil tudi vzrok, da sem se v okviru diplomske naloge hotel posvetiti tematiki, ki bi zajemala različna področja in ne le računalništvo. Pred odločitvijo za študij računalništva in informatike, sem razmišljal tudi o študiju medicine, saj me je to področje že od nekdaj zanimalo. Tudi študijsko prakso sem opravljal v podjetju, ki prav tako deluje na področju medicine. Področje dela mi je bilo zelo zanimivo, zato sem si tudi diplomsko nalogo izbral s tega področja. Z izdelano aplikacijo bom olajšal delo tako zdravnikom, posebej pa tudi pacientom, saj jim ne bo potrebno vedno osebno obiskati zdravnika, ampak bodo lahko svoje bolezensko stanje pogosteje posredovali zdravniku preko izdelane aplikacije. Pojavila se je ideja za izdelavo aplikacije - nadzorne plošče, ki bi pomagala zdravnikom pri spremljanju parkinsonove bolezni. K odločitvi je dodatno prispevala tudi možnost skupnega dela s še dvema diplomantoma in pridobitev dodatnih izkušenj z delom v skupini.

1.2 Cilji

Primarni cilj diplomske naloge je bil razvoj nadzorne plošče, ki bi omogočala zdravnikom lahek pregled pacientov in njihovih podatkov. Zdravniki bodo s pomočjo aplikacije lažje in hitrejše spremljali pojav in napredovanje parkinsonove bolezni.

Aplikacija bo samostojno zajemala podatke, ki jih bo pridobila preko dveh mobilnih aplikacij. Pacienti bodo s pomočjo aplikacije izvajali teste, s katerimi bodo dobili koristne informacije o stopnji in razvoju bolezni ter spremembah v trenutnem stanju bolnikov. Zajete podatke bo aplikacija pretvorila v koristnejšo obliko s pomočjo grafov in prikazom statističnih informacij, ki bodo zdravniku omogočile enostaven pregled stanja za trenutnega pacienta. Prav tako bo omogočena tudi primerjava večih pacientov in pregled širše statistike. Zaradi medicinske narave podatkov je zelo pomemben aspekt varnosti le teh. Tako aplikacija zagotavlja in preprečuje možnost kraje podatkov s strani neavtoriziranih oseb.

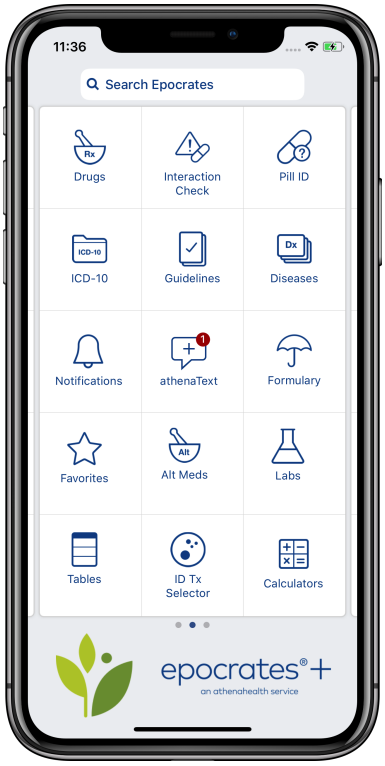
Sekundarni cilj diplomske naloge bo zagotoviti čimboljši uporabniški vmesnik in dodatne funkcionalnosti, ki bi bile koristne zdravniku pri njegovem delu.

1.3 Nadzorna področja (dashboard)

Preden sem začel z izdelavo diplomske naloge, sem se najprej odločil raziskati področja, ki bodo potrebna za izdelavo diplomske naloge. Pri tem se mi je zdelo najbolj koristno, da pregledam nekaj najbolj uspešnih aplikacij, ki so namenjene kot pomoč zdravnikom pri opravljanju njihovega dela. Nato sem na internetu preučil nasvete za izgradnjo čimbolj intuitivne in učinkovite nadzorne plošče.

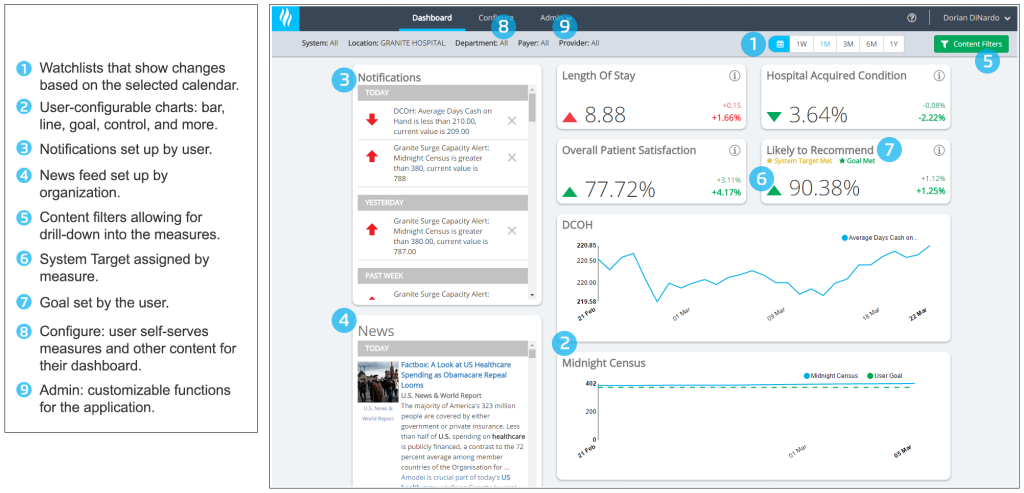
Po pregledu dostopnih informacij sem odkril, da je velika večina aplikacij v svetu medicine prilagojena uporabi na mobilnih napravah. Redke imajo tudi možnost dostopa preko spletnih strani, vendar so le te načeloma veliko slabše izdelane, kot mobilne aplikacije same. Pri pregledu že obstoječih

aplikacij sem ugotovil, da so mobilne aplikacije v veliki večini dostopne tudi osnovnemu uporabniku preko Google trgovine za Android operacijski sistem. Aplikacije, ki so narejene za uporabo na računalniku, niso dostopne za osnovnega uporabnika, temveč le za strokovno uporabo v okviru zdravstva. Zaradi tega sem se odločil, da bom pregledal strukturo mobilnih aplikacij ter se pri njih osredotočil predvsem na razporeditev elementov v njih, njihovem zaporedju in uporabi določenih gradnikov. Pozoren sem bil tudi na to, kako so prikazani različni statistični podatki ter kako zdravnik prejme obvestila. Zaradi velike raznovrstosti aplikacij sem na takšen način dobil boljši pregled praks, ki so se že izkazale kot uspešne in se uporabljalo pri tovrstnih aplikacijah. Lahko sem opazil tudi določene industrijske standarde, ki se uporabljajo pri izgradnji teh aplikacij.



Slika 1.1: Primer ene najbol uporabljenih aplikacij Epocrates.

Naslednji korak je bil pregled nasvetov za gradnjo dobrih nadzornih plošč in uporabniških vmesnikov. Dobra nadzorna plošča naj bi imela jasno in konsistentno poimenovanje vseh svojih elementov, naprimer: merske enote, povezave do drugih strani ali oznake določenih elementov. Barvna shema naj bi bila čimbolj preprosta in konsistentna z že uporabljenimi normami - kot primer: rdeča barva za slabo, zelena za dobro. Število uporabljenih barv naj bo čim manjše, saj je za uporabnika to bolj prijazno. Isto velja tudi za uporabo ikon. Časovni okvirji v različnih grafih naj bodo konsistentni in jasno prikazani. Na enem pogledu se privzete vrednosti časovnih okvirjev ne smejo razlikovati. Formatiranje vrednosti naj bo konsistentno. Prav tako se daljše številčne vrednosti vedno okrajšajo zaradi lažje berljivosti. Za vsak prikaz podatkov je priporočljiva možnost njihovega filtriranja, da lahko s tem dosežemo možnost enostavnejšega prikaza, če si uporabnik to želi. Vsak prikaz podatkov naj bi bil na strani prikazan z določenim namenom. Včasih je lahko manj tudi več. Pri kasnejši izdelavi svoje nadzorne plošče sem se poizkušal čim bolj držati navedenih nasvetov [3].

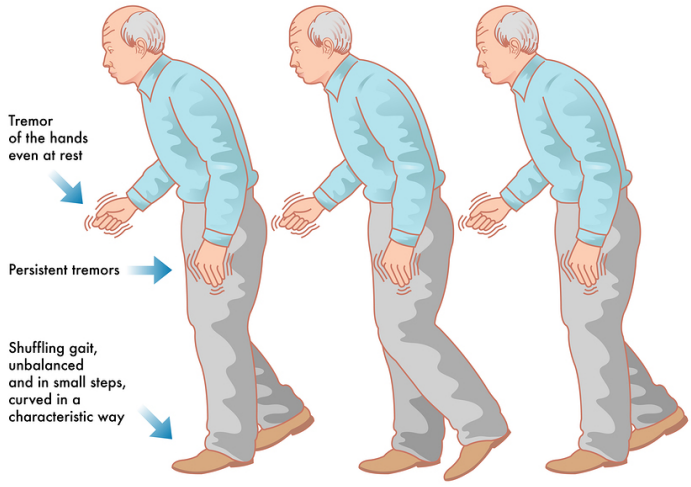


Slika 1.2: Primer učinkovite zdravstvene nadzorne plošče.

1.4 Parkinsonov tremor

Tremor je neprostovoljno tresenje ali tresoči gib. Tipično se pojavlja ob mirovanju in se ponavadi začne v eni od rok ali nog. Kasneje lahko zajame tudi obe strani telesa. Prav tako se lahko pojavi na čeljusti, bradi, ustih ali jeziku. Dodatno lahko nekateri bolnikih s parkinsonovo boleznijo doživljajo občutek notranjega tremorja, ki pa ni opazen na zunaj. Tremor je zelo pogost pojav pri bolnikih s parkinsonovo boleznijo in se pojavi pri približno 80% bolnikov. Bolezen je asimetrična, kar pomeni, da se generalno vedno začne na eni strani telesa. Ta stran telesa bo tudi kasneje ostala bolj prizadeta. Tremor lahko v nekaterih primerih ostane le na eni strani in se ne razširi na celotno telo.

Tremor, ki se pojavlja pri parkinsonovi bolezni, je različen od vseh ostalih tremorjev, saj se načeloma primarno pojavlja le ob mirovanju. Ob premikanju prizatega sklepa tremor izgine, vendar se ob določeni poziciji, ponavadi roke ali prsta, tremor ponovno vrne. To se ponavadi zgodi, ko bolnik v roki drži pribor, zato mu to zelo otežuje življenje [7].



Slika 1.3: Značilne karakteristike parkinsonove bolezni.

Poglavje 2

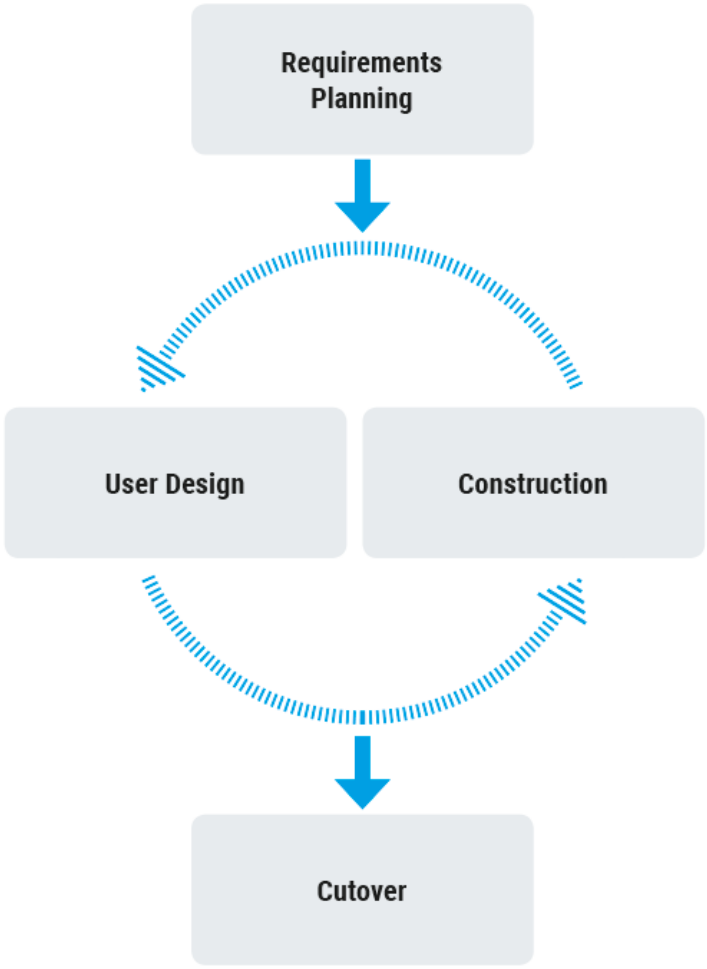
Spletna aplikacija

2.1 Opis razvoja

Pred razvojem aplikacije sem se odločil, da si bom izbral eno od metodologij razvoja programov. Odločil sem se predvsem iz razloga, ker bi lahko na takšen način delal veliko bolj organizirano. S tem bi lahko hitrejše in učinkovitejše našel napake in probleme v svojem načrtu. Zaradi njihove odprave v začetnih fazah načrtovanja bi tako izgubil manj časa pri kasnejši izdelavi aplikacije. Lažje bi si lahko tudi zastavil cilje in časovni okvir, v katerem jih moram doseči. Najprej sem preučil trenutno najbolj uporabljene tehnologije v industriji, ugotovil njihove prednosti in slabosti ter njihov vpliv na razvoj moje aplikacije. Odločil sem se, da zaradi samostojnega dela in ne tako zelo velikega projekta ne bom v popolnosti sledil izbrani metodologiji. Poskušal jo bom čimbolj upoštevati ter si po potrebi kakšen korak tudi prilagoditi.

Pregledal sem več različnih metodologij kot so naprimer: agilna, rapid, waterfall, dinamična, spiralna,... Na koncu sem se odločil za tako imenovani hitri razvoj programov ali RAD. Za ta pristop sem se odločil predvsem zato, ker je ta vrsta procesa najbolj primerna za razvoj programske opreme, kjer je največji poudarek razvoja na uporabniškem vmesniku. Model tudi poudarja sodelovanje končnega uporabnika pri razvoju aplikacije. Na takšen način bi

lahko lažje in hitreje ugotovil pomankljivosti v uporabniškem vmesniku in jih čim lažje odpravil. Ena od slabosti je bilo moje nepoznavanje področja medicine. Zaradi tega bi samostojno težko načrtoval uporabniški vmesnik. Predvsem nisem poznal podatke, kateri imajo največjo dodano vrednosti in kateri ne. Zaradi rednega posvetovanja s končnim uporabnikom bi lahko tudi eliminiral to težavo, saj bi lahko zdravnik redno podajal možne izboljšave s posredovanjem informacij o pomembnosti prikaza posameznih podatkov ter načina, kako naj bodo ti podatki prikazani [6].



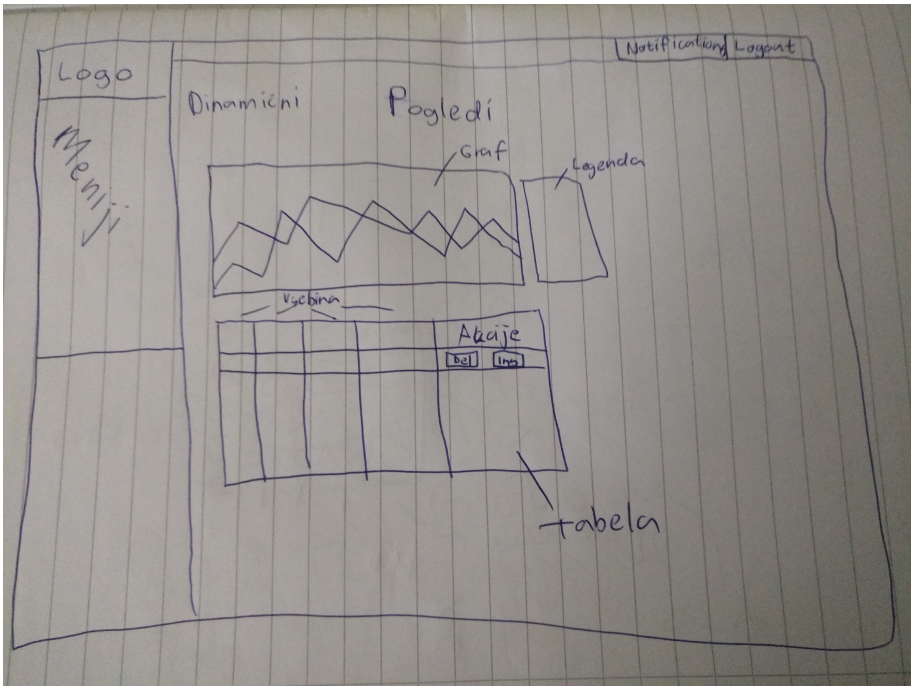
Slika 2.1: Poenostavljen prikaz modela RAD.

2.2 Osnovne zahteve aplikacije

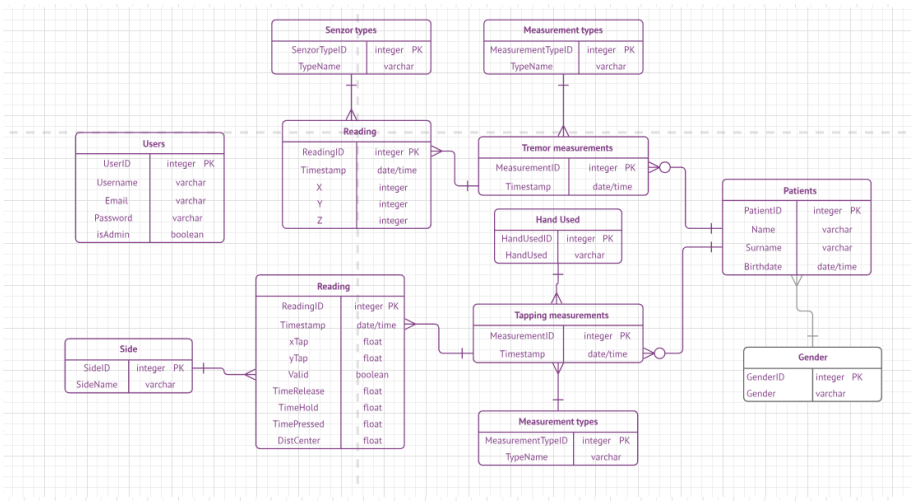
Na začetku izdelave aplikacije sem si najprej poizkušal zapisati vse zahteve, ki jih bo moja aplikacija potrebovala. Na takšen način bi lahko lažje načrtoval njeno delovanje, strukturo, izgled in orodja s katerimi bi delal.
Osnovne zahteve:

- zunanji uporabniki se ne morejo registrirati,
- dva tipa uporabnikov (admin, user),
- avtentikacija uporabnikov pred vstopom,
- baza, ki hrani podatke,
- aplikacijski strežnik,
- možnost zapisovanja podatkov v bazo z mobilnih naprav,
- uporabnik lahko podatke briše, spreminja, vstavlja ,
- aplikacija omogoča več sočasnih uporabnikov,
- zagotovitev varnosti podatkov in dostopa do strani,
- možnost vizualizacije podatkov,
- možnost ogleda surovih podatkov.

Po začrtanju osnovne funkcionalnosti sem se odločil, da naredim še osnovne načrte podatkovne baze in vizualni koncept nadzorne plošče. Ocenil sem, da mi bo to pripomoglo tudi pri izbiri tehnologije, s katero se bom lotil celotnega projekta. Poleg tega si bom lažje predstavljal zahteve in elemente, ki jih bom potreboval. S konceptom baze bom lahko lažje načrtoval potencialne poizvedbe. Pri konceptu postavljanja baze sem imel samo okvirne podatke, ki se bodo beležili na telefonu, vendar pa je bilo to dovolj za njeno izgradnjo.



Slika 2.2: Začetna zasnova grafičnega vmesnika.



Slika 2.3: Začetna zasnova podatkovne baze.

2.3 Uporabljene tehnologije

Po pregledu področja in začrtanju osnovnih zahtev aplikacije, sem dobil dober pregled zahtev, katerim bo morala aplikacija zadostiti. Tako sem lahko veliko lažje izbral tehnologijo s katero bom delal na aplikaciji.

Že pred začetkom dela sem se odločil, da bom back-end programiral v programskem jeziku Java. Odločitev je bila predvsem osebne narave, saj sem ta programski jezik tekom šolanja daleč največ uporabljal. Prav tako sem ga uporabljal tekom prakse in študentskega dela.

V prvem koraku sem izbral bazo s katero bom delal. Odločil sem se za tehnologijo MySQL. Odločitev je bila sprejeta na osnovi ugotovitve, da prednosti pri izboru te tehnologije prevladajo nad njenimi slabostmi. Med prednosti štejem predvsem enostavno uporabo omenjene baze ter dejstvo, da je to eden od industrijskih standardov. Ima tudi zelo veliko bazo uporabnikov, kar mi je omogočalo hitro in lahko iskanje rešitev ob morebitnih nerešenih vprašanjih. Največja slabost uporabe te tehnologije je njena hitrost in določene omejitve pri njeni uporabi. Zaradi predvidene velikosti svoje baze, ki načeloma naj ne bi imela ogromnega števila vnosov ter poizvedb, le te pa bodo dokaj enostavne, se mi ti dve slabosti nista zdeli problematični.

Kot aplikacijski strežnik sem najprej mislil uporabiti JBoss, saj sem ga uporabljal že med študentskim delom na podobnem projektu, vendar sem se kasneje odločil za spletni strežnik Apache Tomcat. To odločitev sem sprejel predvsem na osnovi funkcionalnosti, katere mi je ponujal Tomcat. Zaradi ne tako velike kompleksnosti moje aplikacije bi bila uporaba JBossa, ki sicer ponuja veliko več, popolnoma nepotrebna. O teh dveh možnostih sem tudi primarno razmišljal, ker obe omogočata uporabo tako imenovanih JavaServer Pages, ki omogočajo enostavno kreiranje dinamičnih spletnih strani. Strani temeljijo na osnovi HTML in XML. Podobni so PHP ali ASP jezikom, vendar uporabljajo Javo kot osnovni programski jezik. Ocenil sem, da bom lahko tako s pomočjo JavaServer Pages in Java Servletov popolnoma pokril zahteve svoje aplikacije [5].

Uporabljeno orodje za gradnjo projekta je bil Maven. Zanj sem se odločil

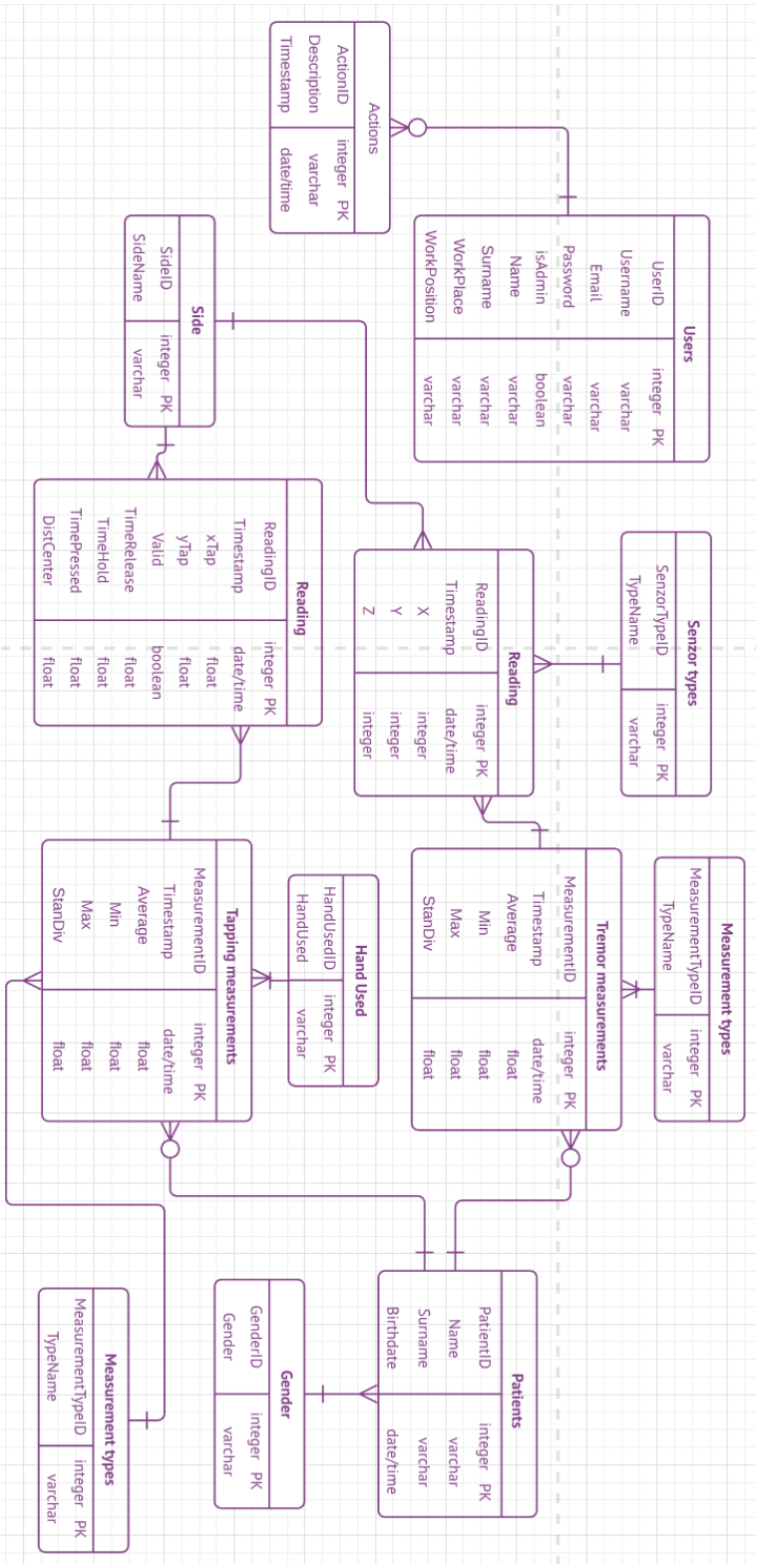
predvsem zato, ker sem se z njim že srečal med študenskim delom in mi je bilo poznano njegovo delovanje. Všeč so mi bile tudi možnosti, ki so dodatno ponujene v primeru, da bi jih potreboval kot naprimer: testiranje, vtičniki, lahko upravljanje razširitev...

Na koncu sem se odločil še za izbiro front-end ogrodja, ki ga bom uporabil za izgradnjo svojega uporabniškega vmesnika. To odločitev sem sprejel, ker se izgradnja strani z uporabo ogrodja zelo poenostavi. Na voljo dobimo že zgrajene komponente, postavitve elementov, barvne sheme, ki so že preizkušene v praksi. Ni nam treba skrbeti za vizualno interakcijo med komponentami in njihovimi barvami, kot bi morali, če bi le te naredili sami. Odločil sem se, da bom uporabil Bootstrap, ki je trenutno najbolj uporabljeno front-end ogrodje na spletu. Na ta način imam zagotavljeno dobro dokumentacijo in veliko podpore zaradi ogromnega števila uporabnikov. Predvsem me je prepričal izgled že narejenih komponent in lahko opravljanje z njihovo postavitvijo. Njihova uporaba bi mi prihranila veliko časa, moja aplikacija pa pridobila na konsistentnosti in lepšem izgledu. Velika prednost je tudi možnost uporabe Javascript pluginov in že narejenih kompleksnejših komponent, kot je naprimer koledar za izbiro datuma. Izbira ogrodja bo zagotovila kompatibilnost v različnih brskalnikih ter med različnimi medsebojnimi prikazi, saj je Bootstrap kompatibilen z vsemi glavnimi brskalniki in zagotavlja konsistenten prikaz med njimi.

2.4 Podatkovna baza

2.4.1 Podatkovni model

Na začetku načrtovanja podatkovne baze sem se odločil, da se bo nahajala v tretji normalni obliki. Odločitev sem sprejel zaradi velikega števila prednosti. Moje mnenje je, da mi bo normalizacija predvsem prinesla hitrost. Za določena polja sem že vnaprej ugotovil, da bodo potrebovala indeksiranje z namenom povečanja hitrosti poizvedb. Zaradi nepodvajanja podatkov bi prihranil tudi prostor. Baza bi s tem postala veliko bolj logično razporejena. Mobilni napravi zaradi tega dejstva tudi nebi potrebovali poznati celotne podatkovne baze, ampak le del, v katerem bi bili shranjeni njuni podatki. Z razvijalci mobilnih aplikacij bi lahko uskladili strukturi obeh delov, kar bi olajšalo delo. Prenos podatkov na mojo podatkovno bazo bi bil veliko lažji zaradi tega, ker pri poizvedbah nebi bilo potrebe po velikem številu joinov (združitev). Tako bi odpadel eden od negativnih aspektov normalizacije.



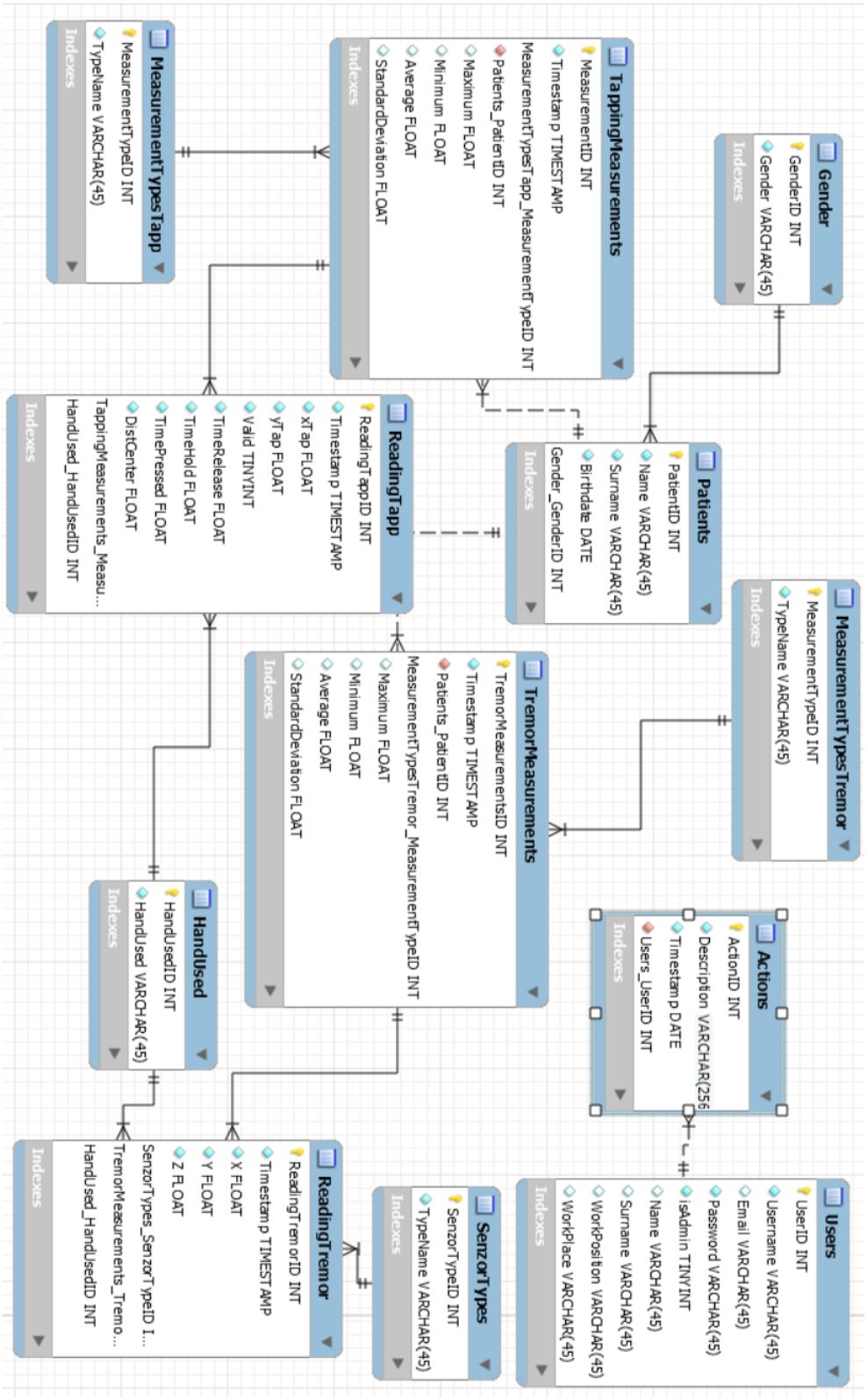
Slika 2.4: Entitetno relacijski model podatkovne baze.

2.4.2 Implementacija podatkovne baze

Preden sem začel z implementacijo baze, sem se še zadnjič posvetoval s kolegoma, ki sta delala na mobilnih aplikacijah. Tako smo skupaj uskadili podatkovne tipe, omejitve in strukturo baze. S tem smo si zagotovili lažje nadaljne delo.

Glede na to, da sem entitetno relacijski model že izdelal, sem se odločil, da bom uporabil za hitrejšo in učinkovitejšo izdelavo baze možnost, ki jo ponuja program MySQL Workbench. Program namreč omogoča kreiranje razširjenega entitetno relacijskega modela in omogoča generiranje SQL skripte iz omenjenega modela za vzpostavitev podatkovne sheme. Na takšen način sem zelo lahko pretvoril svoj že obstoječi model v podatkovno bazo.

Edino dodatno delo je bilo bolj natančno specificiranje podatkovnih tipov atributov in njihovih lastnosti. Preimenovati sem moral tudi nekatere istoimenske elemente, saj tega pri začetnem načrtovanju nisem upošteval. Pri podatkovnih tipih sem le poiskal njihove definicije v MySQL. Definirati sem tudi moral obveznost oziroma neobveznost podatkov.



Slika 2.5: EER diagram podatkovne baze narejen v MySQL Workbench.

2.4.3 Poizvedbe in delo s podatki

Po pregledu baze in poizvedb sem bil mnenja, da dodatnih indeksov poleg primarnih in tujih ključev ne potrebujem, saj se zaradi narave poizvedb le te ne bi veliko pohitrile.

Razmislil sem tudi, kakšne akcije bodo uporabniki lahko izvajali na podatkih in kako bi to lahko vplivalo na njih. Ugotovil sem, da je edina akcija, ki bi lahko na moji bazi povzročala probleme brisanje posameznih podatkov. Ker lahko uporabniki brišejo le lastne podatke, dodatno administratorski računi pa še podatke o uporabnikih, sem moral preučiti le ta dva primera. Ugotovil sem, da je brisanje računov problematično, saj se lahko njegov primarni ključ navezuje na druge tabele. Brisanje uporabnikov predstavlja težavo zaradi ostalih vnosov, ki se navezujejo nanj.

Za potrebe brisanja računov sem tako naredil trigger, ki ob primeru brisanja, najprej izbriše vse vnose akcij v tabeli actions, ki jih je uporabnik do sedaj že naredil, potem se izbriše še uporabnik sam. Tako razrešimo problem z navezovanjem na primarni ključ v tabeli users.

Najti sem moral rešitev, kaj bom naredil v primeru, da administrator iz podatkovne baze zbriše enega od pacientov, želi pa obdržati njegove meritve. Za ta problem obstaja veliko različnih rešitev. Osebno sem se odločil, da bi bila najboljša rešitev za ta problem anonimizacija podatkov. Za to sem se odločil, ker lahko podatke ohranimo in jih še naprej uporabljamo za določene statistične obdelave, kljub temu, da podatki niso več povezovani z določenim pacientom.

Odločiti sem se moral za rešitev, kako izvesti anonimizacijo. Primerna rešitev mojega primera je, da se pacientov vnos kljub izbrisu ohrani v obliki primerni za statistično obdelavo. Ime in priimek pacienta bi spremenil na anonimen. Takšni računi v uporabniškem vmesniku ne bi bili vidni. S tem bi na zunaj pacient deloval izbrisan in tudi v bazi ne bi mogli več najti točnega podatka o njem. Vseeno bi imeli dostop do vseh njegovih dosedanjih meritev, ki bi bile vezane nanj. To nam bi lahko pomagalo pri kasnejših statistikah ter analizah ostalih pacientov in strojnem učenju. Na voljo bi imeli vedno

vse meritve, ki so se kadarkoli opravile na naših aplikacijah in podatek o tem, katere meritve so bile opravljene s strani določenega pacienta.

2.5 Varnost in avtentikacija

Odločil sem se, da bo avtentikacija uporabnika strani potekala preko strani za vpis. Na strani uporabnik vnese svoje podatke in s tem potrdi svojo pristnost. Ob pravilnem vnosu podatkov se uporabnika nato preusmeri na začetno stran nadzorne plošče.

2.5.1 Prijava in odjava

Podatke o uporabnikih, kot so uporabniško ime, elektronski naslov in geslo, sem se odločil hraniti v svoji podatkovni bazi. Ob poskusu vpisa uporabnika, se tako uporabnikovi podatki primerjajo s tistimi shranjenimi v podatkovni bazi. V primeru ujemanja podatkov, se uporabnikova identiteta potrdi in izvede se vpis uporabnika v aplikacijo. V primeru napačnega vnosa podatkov se uporabniku sporoči, da je napisal napačno uporabniško ime ali geslo. Vstop pa mu je zavrnjen. V primeru uspešnega vpisa uporabnika se tudi nastavijo na HTTP seji trije atributi: ti so uporabniško ime, uporabnikov ID in podatek o tem ali je uporabnikov račun administratorski ali ne, ker je le ta potreben za kasnejše delo. Uporabnik pa je preusmerjen na začetno stran nadzorne plošče.

Ob izpisu uporabnika iz spletne strani je potrebno narediti že omenjene korake v obratnem vrstnem redu. Najprej se odstranijo vsi atributi seje, saj s tem preprečimo že izpisanemu uporabniku dostop nazaj do strani, ki jih je že prej obiskal. Uporabnika pa preusmerimo na stran za vpis.

2.5.2 Preprečitev predpomnenja

Želel sem tudi preprečiti že izpisanemu uporabniku dostop do strani, ki jih je že obiskal preko gumba nazaj v brskalniku. To sem dosegel preko kontro-

liranja predpomnilnika brskalnikov. Da bi lahko to storil preko vseh najbolj uporabljenih brskalnikov, sem potreboval naslednji ukaz:

```
response.setHeader("Cache-Control", "no-cache, no-store,
    must-revalidate");
```

Parameter no-store pove brskalniku, da v svoj predpomnilnik ne bo hranil ničesar o klientovi zahtevi ali odgovoru strežnika. No-cache pove, da brskalnik ne bo serviral uporabniku shranjene verzije v predpomnilnik, brez da bi strežnik o tem prej povprašal. Se pravi, da bo to verzijo predložil le v primeru, če se originalna na strežniku ni spremenila. Must-revalidate pove, da bo brskalnik vedno preveril vsebino, ki je shranjena v predpomnilniku. V primeru da je zastarela, je ne bo uporabil. Na takšen način zagotovimo, da se vsebina zagotovo ne bo shranila v kateremkoli od glavnih brskalnikov.

2.5.3 Preprečitev vstopa brez avthentikacije

Ker nisem želel, da bi lahko uporabniki preko URL naslovov dostopali do strani, ki so jih predhodno že obiskali, brez da so vpisani, sem moral zagotoviti, da strežnik takšno poizvedbo zavrne. To sem storil s pomočjo JSP filtrov. Le ti se uporabljajo za prestrezanje zahtev uporabnikov, preden ti dostopijo do virov. Za tem manipulirajo strežnikov odgovor, ki se ga pošlje uporabniku.

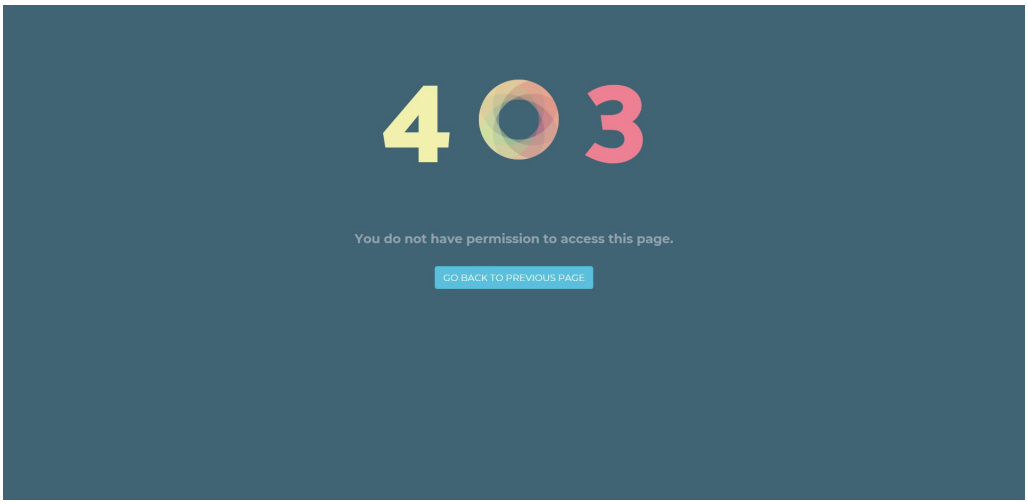
Naredil sem filter, ki najprej preveri, če ima seja shranjen podatek o uporabniku. V primeru da je temu tako, se je uporabnik že prijavil v aplikacijo in je dostop do vseh linkov omogočen. V nasprotnem primeru je uporabnik, ne glede na to, kateri URL hoče odpreti, vedno preusmerjen na začetno stran za prijavo. Tako lahko uporabnik dostopa brez prijave le do začetne strani. Dostop do ostalih pa mu je s tem preprečen.

Strežnik filtrira samo naslove z URL vzorcem /content/*. S tem tudi omogočimo lahko dodajanje nove vsebine, ki jo ali pa je ne želimo filtrirati. V primeru, da si tega želimo, se mora nova vsebina nahajati v mapi content. S tem se filter avtomatsko uporabi. V nasprotnem primeru svoj vir

pozicioniramo kje drugje. Na ta način se filter ne bo uporabil.

2.5.4 Omejitev dostopa do določenih virov

Kot sem moral omejiti dostop do virov zunanjim uporabnikom, ki se v aplikacijo še niso prijavili, sem moral to storiti tudi znotraj aplikacije same. V aplikaciji sem moral omejiti dostop do administrativnih virov, se pravi tistih, s katerimi upravljamo podatke o uporabnikih aplikacije. Odločil sem se za enak pristop kot prej. Omenjene vire sem premaknil v mapo admin, ob poskusu dostopa do nje pa sem preveril, ali je trenutno prijavljen račun administratorski ali ne. Glede na to sem dostop do strani omogočil ali pa zavrnil. Odločiti sem se moral, kako bi uporabniku sporočil, da do te strani nima dostopa.



Slika 2.6: Stran na katero naleti osnoven uporabnik, če dostopi do administratorskih strani.

Pri dostopanju virov znotraj aplikacije, brez da bi se uporabnik vpisal, se mi ni zdelo problematično, da je le ta preusmerjen na stran za vpis. V tem primeru je to zadostna informacija za uporabnika. Če bi to storil ponovno, uporabnik ne bi vedel, zakaj točno do vira ni mogel dostopiti in zakaj je bil preusmerjen. Zato sem se odločil, da bom uporabnika v tem primeru

preusmeril na stran z HTTP 403 napako. Napaka 403 je standarni status, ki se kot informacija prenese do uporabnika. Pove mu, da je strežnik razumel zahtevo, vendar je ne bo izpolnil. Za razliko od napake 401, ki nam sporoča, da do vira ne moremo dostopati, ker se moramo še dodatno avtenticirati. Napaka 403 nam sporoči, da do vira kljub avtentikaciji ne moremo dostopiti, ker nam dostop do vira ni dovoljen [1]. Zato sem bil mnenja, da je to status, ki je najbolj primeren za prikaz uporabniku.

2.5.5 Varne povezave

Zaradi občutljive narave podatkov sem moral zagotoviti tudi varne povezave. To sem naredil s pomočjo uporabe certifikata, ki ga strežnik uporabi za vzpostavitev varne povezave z uporabnikom. Varna povezava je omogočena na vseh straneh spletne aplikacije. Uporabljen je kriptografski protokol TLS 1.2.

2.6 Uporabniški vmesnik

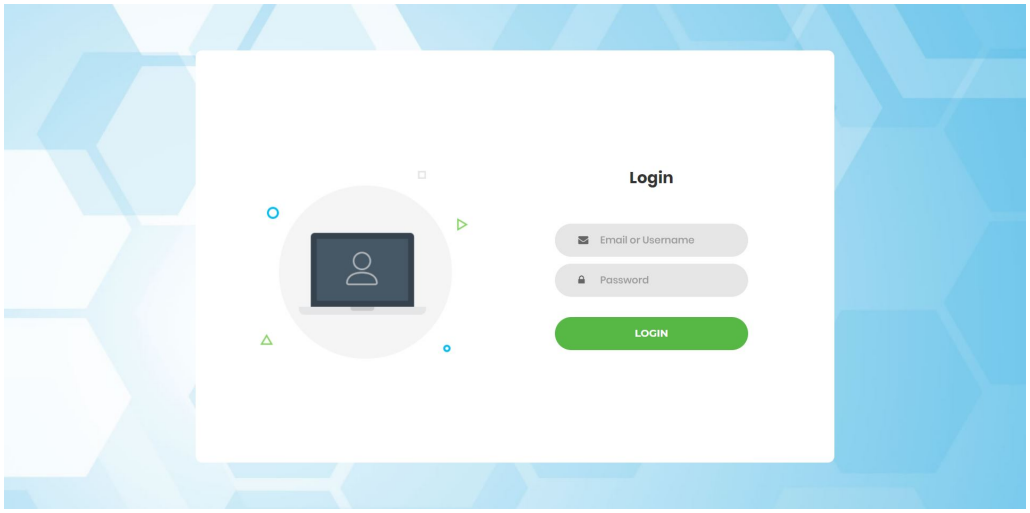
Odločil sem se, da bo uporabniški vmesnik narejen generalno v modrikasti barvi, saj sem mnenja, da ljudje pri tej barvi dobijo asociacijo na medicino. Kot primer lahko vzamemo tudi spletne strani Univerzitetnega kliničnega centra Ljubljana in spletno stran podjetja Lek. Pri obeh lahko opazimo modro barvno temo strani, kar se tudi odraža v njihovih logotipih [2].

Slika 2.7: Primer modre barvne teme na spletni strani UKC Ljubljana.

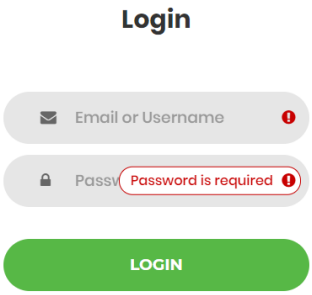
2.6.1 Vpisna stran

Za ozadje vpisne spletne strani sem izbral čimbolj enostavno sliko z modro barvno shemo zato, da le ta ne bi uporabnika odvrnila od pomembnejših elementov na strani. Z uporabo standardnih znakov sem uporabniku tako besedno, kot tudi slikovno ponazoril, kje se nahajajo polja za vpis. Na takšen način uporabnik hitro dobi asociacijo in se njegova pozornost preusmeri na njih. Gumb za vpis je zelene barve, saj je namreč to barva, ki je v barvni teoriji komplementarna modri. Desno od vpisnega okna sem dodal tudi sliko, ki ponazarja vpis uporabnika. Tako sem zapolnil praznino na spletni strani in naredil ločnico med podobnostjo z mobilnim vpisom.

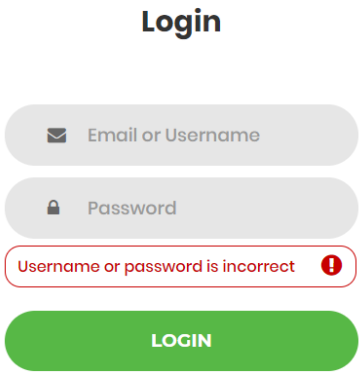
Moj uporabniški vmesnik je uporabnika moral opozoriti tudi ob nepravilnih akcijah in kako jih popraviti. Na vstopni strani je težava lahko nastala pri vnosu napačnih podatkov ali če uporabnik ne vnese vseh podatkov. To sem storil z opozorili rdeče barve, ki se ponavadi uporabljajo za napake na mestih, kjer se je napake zgodila. Tako naj bi uporabnik tudi brez brez branja opozorila čimpreje zaznal, kje se nahaja težava.



Slika 2.8: Začetna vpisna spletna stran.



Slika 2.9: Opozorilo, ki ga uporabnik prejme, ko podatkov ni vnesel.



Slika 2.10: Opozorilo, ki ga uporabnik prejme, če je vnesel napačne podatke.

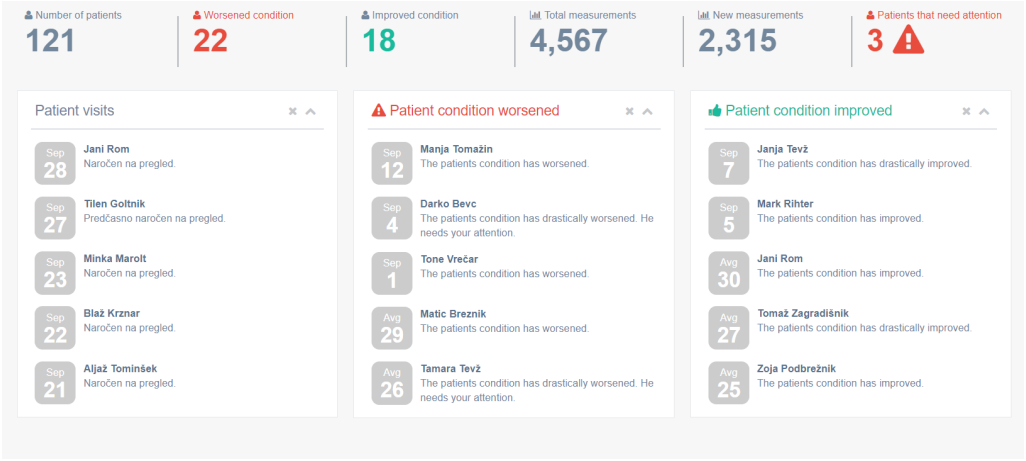
2.6.2 Začetna stran

Na začetni strani nadzorne plošče sem uporabniku poskušal podati čimveč generalnih informacij o meritvah samih in o pacientih. Stran je zaradi pomankanja umetne inteligence v tem trenutku samo ohranitvenik. Predstavlja pa osnovni namen, ki ga naj bi začetna stran dosegla.

Zdravnika bi naj opozorila o trenutnem stanju pacientov in mu podala generalno informacijo o meritvah samih. Dobil bi tudi podatek o izboljšanju ali poslabšanju stanja določenih pacientov ter o njihovih naročilih za obisk zdravnika. Po mojem osebnem mnenju so to informacije, ki so najbol koristne in pomembne. Zdravnik pa jih mora videti čimprej, ko se prijavi v spletno stran. Za vse bolj podrobne informacije o pacientih ali meritvah ima na voljo specializirane strani.

Pri izgradnji strani sem uporabnika do pomembnejših podatkov poskušal usmeriti s pomočjo barv in ikon. Tako so negativni oziroma zelo pomembni

podatki obarvani rdeče. Pozitivni pa z zeleno barvo. Z ikono v velikosti besedila sem tudi pripomogel, da uporabnik takoj fokusira svojo pozornost nanjo, saj je tudi s tem najbolj pomembna informacija na celotni strani. Ker si sporočanje dogodkov sledi v časovnem zaporedju, sem se odločil, da bom element naredil v koledarski obliki, ki je večini uporabnikov domača. Tako uporabniku sporočimo kdaj se je, ali pa se še bo, določen dogodek pripetil. Podamo pa mu tudi informacijo o vsebini dogodka, ki se je zgodil in pripadajoči osebi.



Slika 2.11: Izgled vsebine začetna strani na nadzorni plošči.

2.6.3 Stranski meni

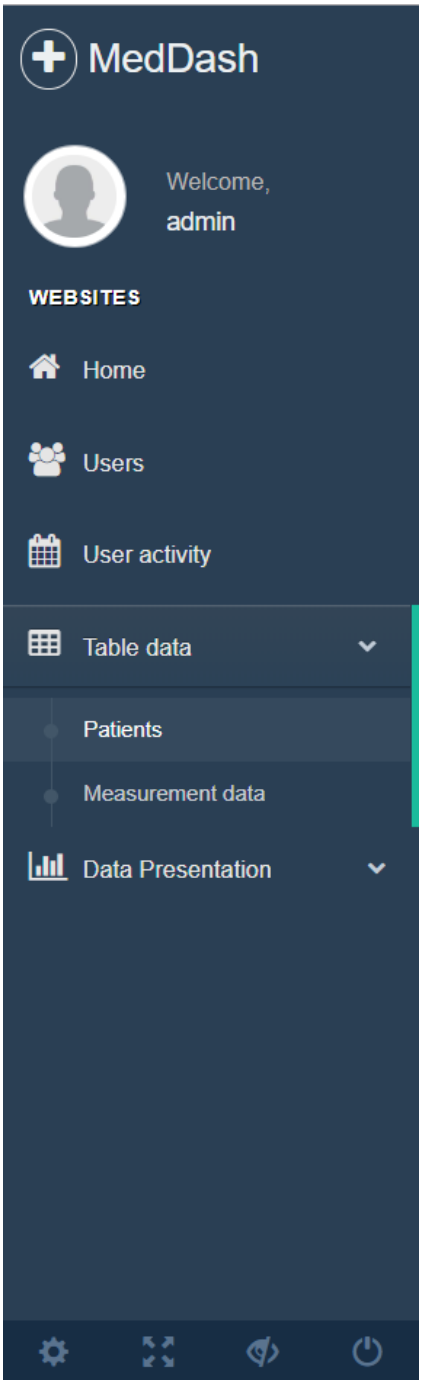
Odločitev, da se bo uporabnik med stranmi pomikal z uporabo stranskega menija sem sprejel zato, ker je takšen način postavitve uporabljen v veliki večini spletnih nadzornih plošč. Osebno mi vertikalna postavitev vizualno bolj ugaja kot vodoravna.

V zgornji desni kot sem postavil ime aplikacije in pa simbol, ki ponazarja, da je osnovni namen aplikacije medicinske narave. Pod njim se nahaja vrstica s podatki o uporabniku. S klikom na sliko uporabnik dostopi do svojega profila. Nato sledi seznam strani, do katerih lahko uporabnik dostopa. Stran

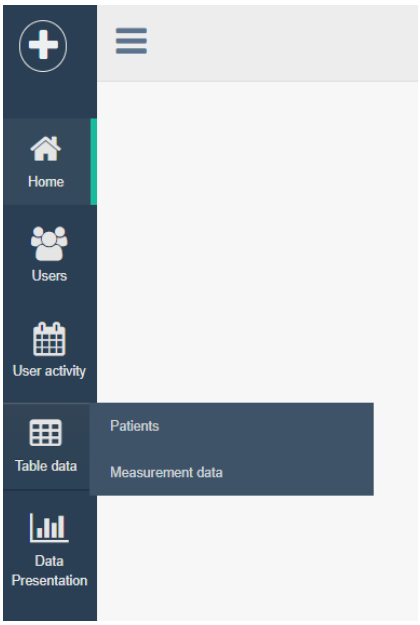
na kateri se trenutno nahajamo je označena s prehodom barve na svetlejši odtenek. Desna zelena črta nakazuje mesto menija v katerem se nahajamo.

V spodnjem delu razširjenega menija sem dodal tudi štiri bližnjice, ki so se mi zdele koristne za uporabnika. Prva omogoči uporabniku hiter dostop do nastavitev profila. Druga razširi aplikacijo na celoten zaslon, kar je lahko koristno pri opazovanju večjih tabel ali grafov. Tretja zmanjša stranski meni na minimalno velikost. Četrta uporabnika izpiše iz aplikacije. Namen njihove uporabe sem poizkušal čimbolj ponazoriti z ikonami, ki sem jih uporabil. Poleg tega sem dodal tudi besedni opis, ki se pojavi v primeru, če uporabnik lebdi s kurzorjem nad eni izmed ikon.

Ker je lahko velik stranski meni tudi ovira, lahko uporabnik slednjega tudi zmanjša. To ga naredi veliko bolj kompaktnega za uporabo, saj je uporabniku viden le še logo aplikacije in strani do katerih lahko dostopa. V primeru gnezdenega seznama se uporabniku ob kliku nanj na desni strani pojavijo strani, ki jih ta seznam vsebuje. Velikost stranskega menija se ne spremeni.



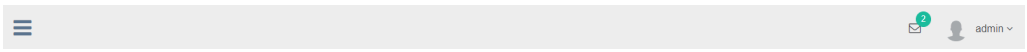
Slika 2.12: figure
Razširjen stranski meni.



Slika 2.13: figure
Minimiziran stranski meni.

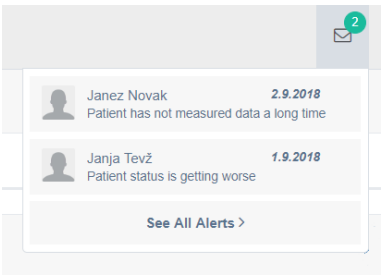
2.6.4 Glavni/zgornji meni

Kot je bil potreben meni za izbiro spletne strani, na kateri se nahajamo, je moja nadzorna plošča potrebovala tudi mesto, kjer bo uporabnik lahko dostopil do svojega uporabniškega računa. Najbolj se mi je zdela primerna uporaba vrstice na vrhu aplikacije, saj se tega poslužujejo tudi številne druge strani. Tako se na levi strani omenjene vrste nahaja gumb, ki omogoči povečanje in zmanjšanje stranskega menija. Na desni strani imamo zavihek z obvestili in zavihek, ki nam omogoča dostop do strani povezanih z uporabniškim računom.



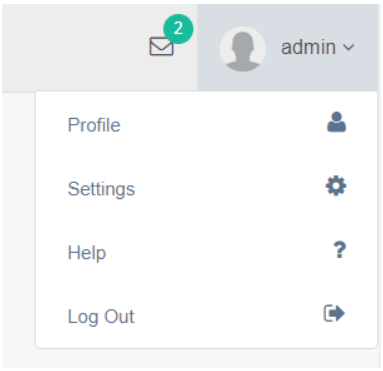
Slika 2.14: Postavitev zgornjega menija.

Zavihka z obvestili na žalost zaradi časovne omejitve nisem uspel funkcijsko usposobiti, saj bi za njegovo delovanje potrebovali umetno inteligenco, ki bi analizirala meritve pacientov in administratorja - zdravnika opozarjala na nepravilnosti. Zato je na strani prikazan le ohranitvenik, ki bo zdravniku posredoval število novih obvestil, ki jih je prejel. Poleg tega bo prejel informacijo za katerega pacienta gre, njegovo ime ter kdaj in kakšna je vsebina generiranega sporočila.



Slika 2.15: Opozorila, ki jih uporabnik prejme s strani aplikacije.

V meniju uporabniškega računa se nahajajo štirje zavihki, s katerimi lahko dostopamo do uporabniškega profila ali nastavitev zanj, dobimo pomoč pri uporabi strani, ali pa se izpišemo iz svojega računa. Ikone sem pozicioniral na desni strani. To sem storil zato, ker sem bil mnenja, da so vidne prej, saj se tako nahajajo pod menijem. Takšna izbira je boljša, ker uporabnik tako hitreje dobi asociacijo glede uporabnosti menija.



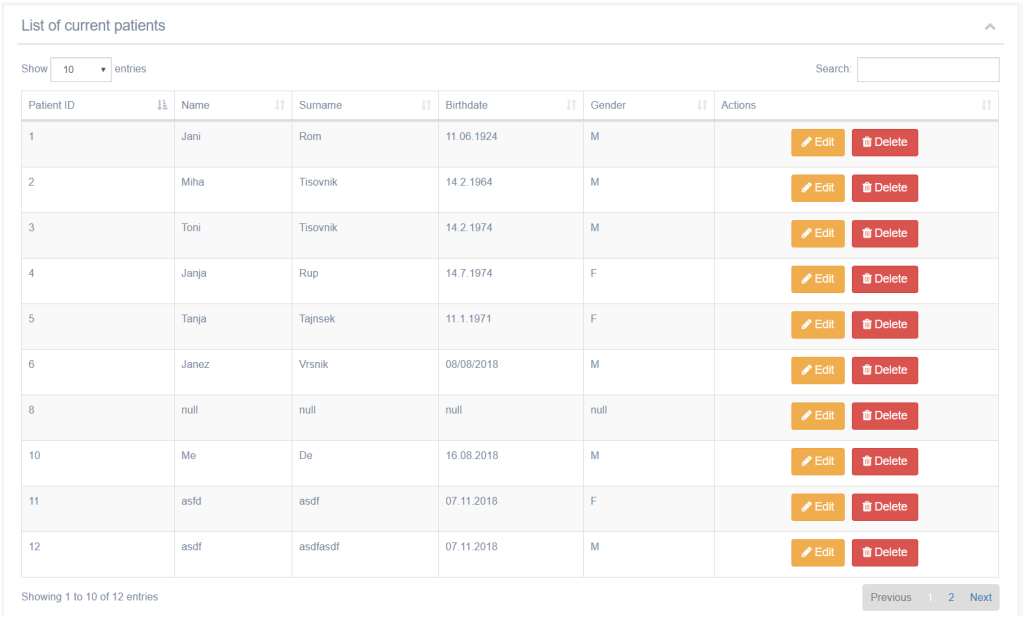
Slika 2.16: Meni povezan z uporabnikovim profilom.

2.6.5 Tabele

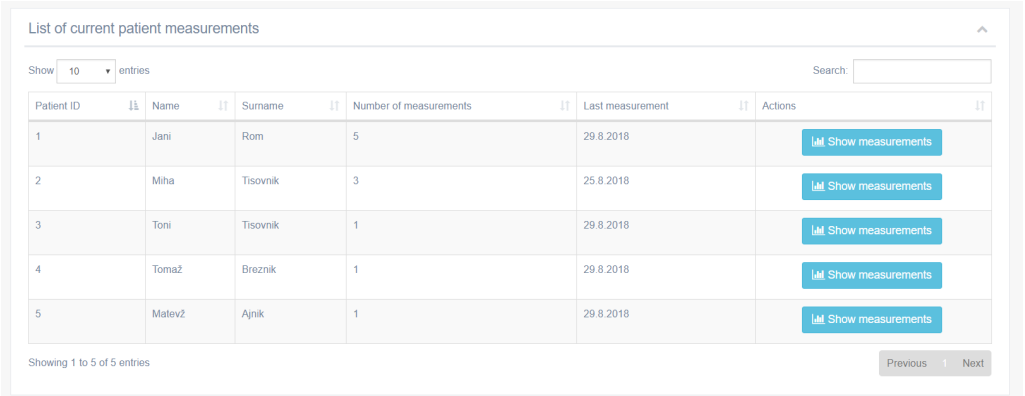
Pri izdelavi tabel sem se spopadal s problemom možnega kasnejšega velikega števila vpisov ter kako jih učinkovito prikazati v tabeli. Odločil sem se, da bom ta problem rešil z dvema pripomočkoma. Nad vsako tabelo se tako nahaja možnost, s katero lahko omejimo število zapisov, ki se nam prikažejo na eni strani. Poleg tega lahko v vnosnem polju filtriramo zapise glede na poljuben stolpec v tabeli. Na dnu tabele se nam poda tudi informacija o tem, katere zapise nam tabela trenutno prikazuje. Imamo pa tudi možnost pomikanja med stranmi v tabeli.

Tabela sama ima omogočeno urejanje vrstic glede na poljuben stolpec, tako naraščajoče kot tudi padajoče glede na vrednosti v tabeli. Vrednosti so lahko številčne ali besedne. Vse tabele sledijo enakemu formatu. Na levi strani se nahajajo podatki o vrstici, na desni strani akcije, ki jih lahko

počnemo z omenjeno vrstico. Te akcije so tudi primerno obarvane, kjer rdeča barva ponazarja brisanje podatkov, oranžna ponazarja urejanje, modra barva pa prikaz večih informacij o izbrani vrstici.



Slika 2.17: Tabela, namenjena urejanju podatkov.

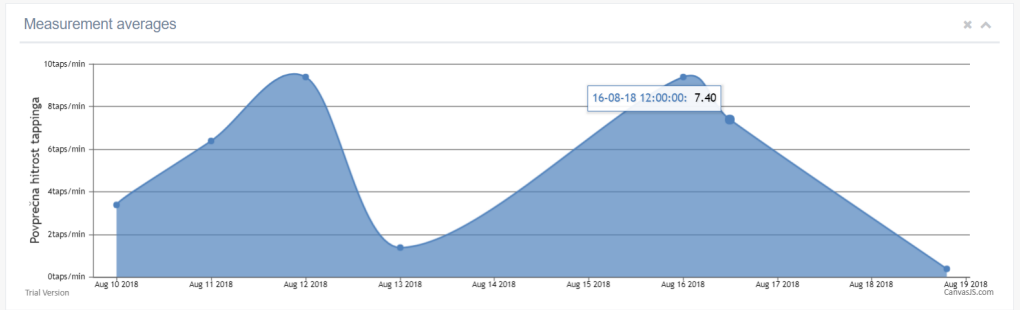


Slika 2.18: Tabela, namenjena ogledu podatkov.

2.6.6 Vizualna predstavitev

Pri grafični predstavitvi podatkov sem se odločil za uporabo knjižnice CanvasJS. Izbral sem jo, ker sem bil mnenja, da je zagotavljala najbol intuitivno predstavitev podatkov in vse funkcionalnosti, ki jih bomo potrebovali pri nadaljnem delu. Ob primeru nadaljnih nadgradenj bodo le te zadostne in ne bo potrebno iskati boljše alternative.

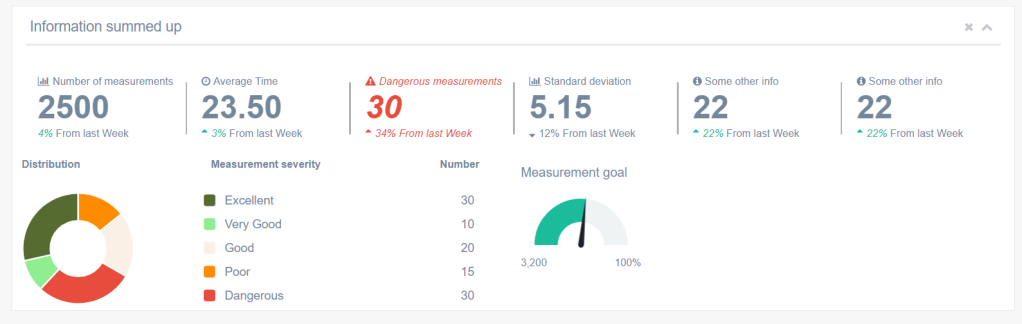
Velika prednost, ki jo ima knjižnica nasproti ostalimi, ki sem jih tudi potencialno preveril, je odličen prikaz X osi za časovni potek. Kar je zelo pomembno glede na to, da bo načeloma pri vseh grafih X os takšne oblike. Graf nam omogoči približevanje in pomikanje po X osi. Ob primeru približevanje se skala X osi tudi spremeni. Se pravi, da v primeru, ko dovolj približamo, enote na X skali spremenijo obliko iz datuma v uro. To nam omogoči veliko lažji pregled meritev. Poleg tega pa ob primeru lebdenja z miško nad meritvijo dobimo točen podatek o njej, kot tudi podatek o tem, kdaj je bila narejena.



Slika 2.19: Grafična predstavitev podatkov.

Naredil sem tudi element, ki naj bi uporabniku predstavil povzetek informacij meritev o določenem pacientu. Element naj bi podal čimveč informacij o meritvah samih, kot je naprimer njihova ocena ob razvoju umetne inteligence. Kako pogosto so opravljane, kako zelo se razlikujejo, kako hitro so opravljene, kot tudi sprememba teh podatkov v določenem časovnem obdobju. Element je v tem trenutku le ohranitvenik in se pri vseh pacientih kaže

v istem formatu. V končni obliki bi se mogoče spremenili podatki, ki jih slednji element prikazuje, format in oblika pa bi ostala enaka.



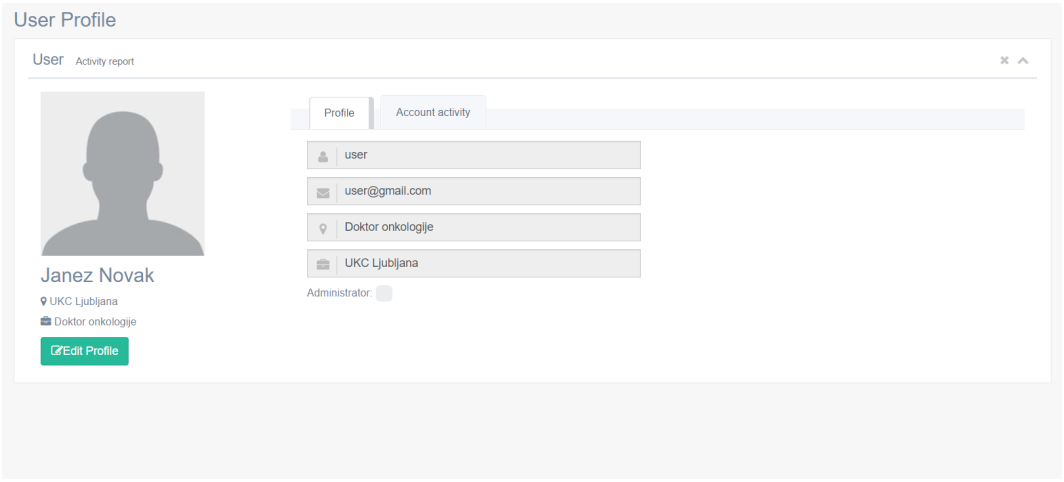
Slika 2.20: Povzetek informacij o pacientovih meritvah.

2.6.7 Profilna stran

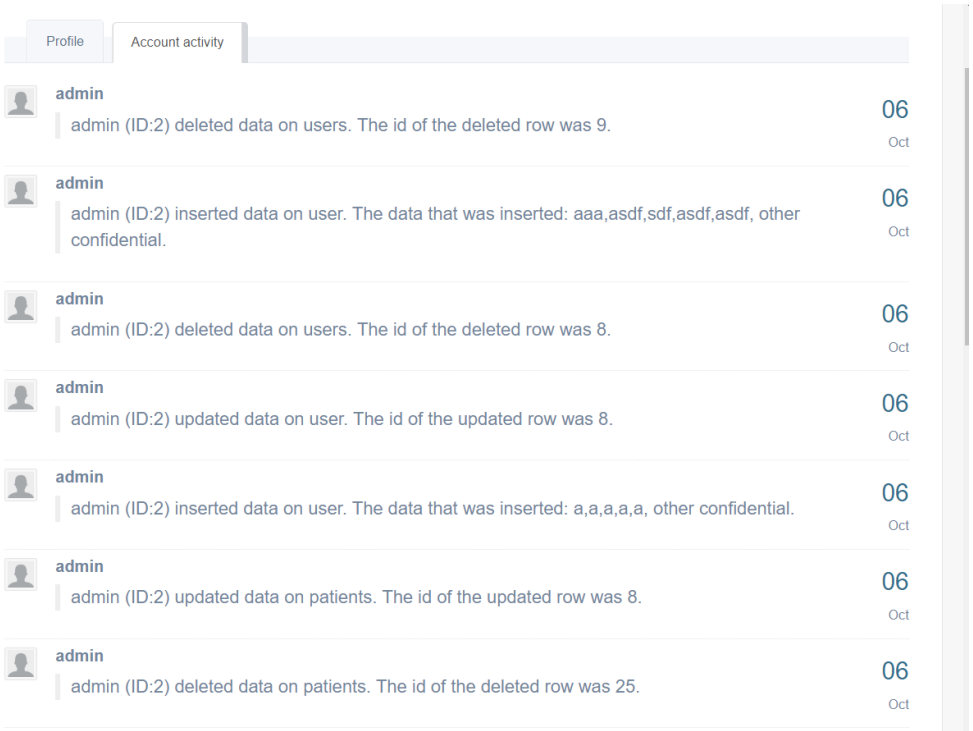
Uporabnik lahko na profilni strani preveri vse svoje podatke o profilu, prav tako lahko vidi tudi vse izvedene aktivnosti (brisanje, vstavljanje in spreminjanje podatkov v tabelah).

Na začetni strani dobi uporabnik najprej vse podatke o svojem računu. Te lahko tudi spreminja s klikom na gumb, ki se nahaja pod profilno sliko.

Če uporabnik izbere zavihek za aktivnost, lahko pregleda zadnjih 20 dejanj, ki jih je naredil. Pri tem dobi informacijo o aktivnosti, ki jo je izvedel, na kateri tabeli, v kateri vrstici v tabeli, kakor tudi datum izvedbe aktivnosti.



Slika 2.21: Profilna stran.

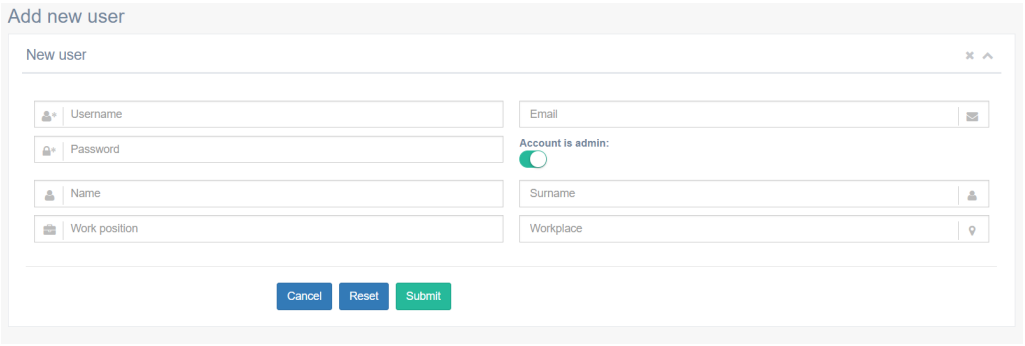


Slika 2.22: Aktivnosti računa na profilni strani.

2.6.8 Obrazci za urejanje podatkov

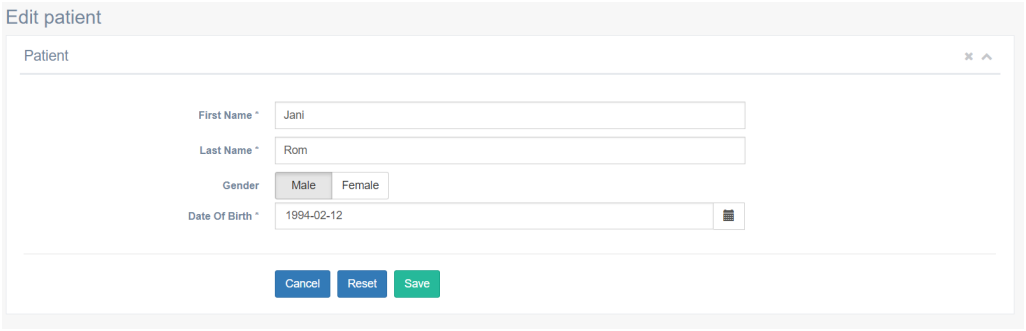
Za potrebe dodajanja in urejanje že obstoječih vnosov v tabeli sem naredil obrazce za vnos. Pri vseh obrazcih so obvezna polja označena z *. Vsa polja pa imajo vizualne namige preko uporabe ikon za hitrejšo prepoznavo namena. Za vnos navadnih nizov sem uporabil vnosna vpolja tipa tekst. Za vnos gesla pa vnosno polje tipa password. Za izbiro vrednosti tipa boolean sem uporabil ali potrditveno polje ali radijski gumb.

Pri obrazcu za vnos novega polja so vsa polja brez vrednosti in prazna. Na mestu vrednosti se nahajajo ohranitveniki, ki uporabniku besedno povedo, čemu je polje namenjeno.



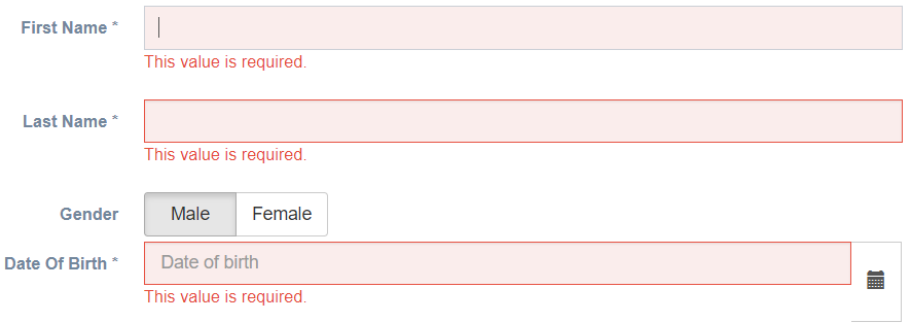
Slika 2.23: Obrazec za dodajanje novega polja v tabelo.

V primeru spreminjana določenega vnosa, se vse vrednosti vnosnih polj nastavijo na trenutne vrednosti le tega. Oblika obrazca pa ostane enaka.

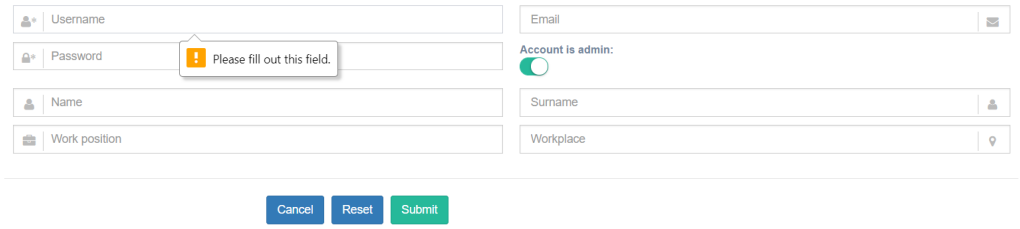


Slika 2.24: Obrazec za urejanje obstoječega polja v tabeli.

Glede na to, da so nekatere vrednosti v tabelah obvezne, nas obrazci tudi sami obvestijo o tem, da jih nismo vnesli, preden se transakcija sploh pošlje v podatkovno bazo. Zaradi dveh različnih struktur obrazcev sta narejena dva različna načina sporočanja. Tako se nam v primeru, kjer je potrebno večje število vnosov različnih podatkov, struktura obrazca ne podere. Vendar je negativni aspekt ta, da nam uporabniški vmesnik ne sporoči vseh napak naenkrat.



Slika 2.25: Zaradi manjšega števila vnosnih polja nas obrazec opozori na vse nepravilnosti.



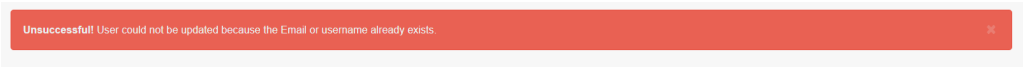
Slika 2.26: Zaradi večjega števila vnosnih polj nas obrazec opozori na prvo nepravilnost.

2.7 Povezava na bazo

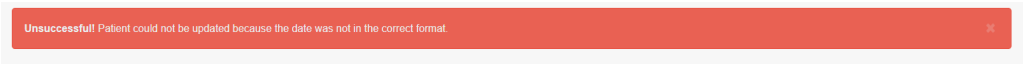
Podatke za spletno stran dinamično pridobivam s povezavo na podatkovno bazo prek programskega jezika Java. To poteka ali v servletih ali kar v JSP samem. Povezava sama poteka preko SSL povezave. Tako so podatki zavarovani tudi pri prenosu med podatkovno bazo in končnim uporabnikom.

2.7.1 Preverjanje pravilnosti podatkov

Pravilnost podatkov preverjam pred fazo zapisovanja. Uporabniški vmesnik mi že sam zagotovi, da so vsa vnosna polja vnešena. Backend mora poskrbeti le še za preverjanje pravilnosti podatkov. Vedno preverim pravilno obliko datuma, saj kljub koledarju za izbiro datuma, lahko uporabnik tudi sam vnese datum v nepravilni obliki. Pri vnašanju novih pacientov v podatkovno bazo načeloma ne more priti do zapletov, saj lahko za vnos sprejmemo karkoli. Pri dodajanju novega uporabnika se administratorja opozori, če se enako uporabniško ime ali elektronski naslov že nahaja v podatkovni bazi.

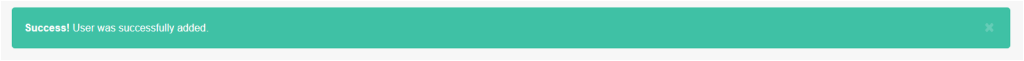


Slika 2.27: Opozorilo, ki ga uporabnik prejme ob dodajanju že obstoječega uporabnika.

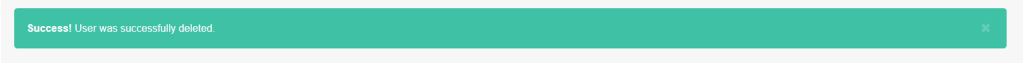


Slika 2.28: Opozorilo, ki ga uporabnik prejme ob vnosu napačne oblike datuma.

Uporabnika opozorimo na nepravilnost ob vnosu, prav tako pa ga tudi opozorimo na uspešno zaključene transakcije. Tako se uporabniku ob uspešno zaključeni transakciji prikaže enaka oblika sporočila vendar v zeleni barvi.



Slika 2.29: Primer sporočila, ki ga uporabnik prejme ob uspešni transakciji dodajanja uporabnika.



Slika 2.30: Primer sporočila, ki ga uporabnik prejme ob uspešni transakciji odstranjevanja uporabnika.

2.7.2 Problem kodiranja

Pri prenosu podatkov s strani odjemalca na strežnik se je pojavila težava kljub temu, da sem imel spletne strani kodirane v obliki UTF-8, podatki pa so se na strežnik prenašali v obliki kodiranja CP-1252. Tukaj je prišlo do težave zaradi nepravilnega prikaza šumnikov in posledičnega nepravilnega shranjevanja podatkov v bazo. Kljub temu, da je bila stran sposobna prikazovati, nismo mogli z njimi operirati. Problem sem rešil z uporabo filtra, ki se izvaja na vseh straneh. Njegova naloga je, da pri vsaki zahtevi uporabi UTF-8 kodiranje in jo enostavno posreduje naprej. Tako sem omogočil tudi shranjevanje šumnikov in ostalih znakov v podatkovno bazo.

2.7.3 Beleženje aktivnosti

Aplikacija beleži tudi aktivnosti uporabnikov v podatkovni bazi. Tako lahko vemo kateri uporabnik, kdaj in kaj je spreminjal v podatkovni bazi. Pri tem beležimo le vstavljanje, brisanje in spreminjanje baze, ne pa poizvedb samih, saj podatek o tem ni pomemben. Vsakič, ko se naredi sprememba podatkov v podatkovni bazi, se takoj vanjo zapiše podatek o tem kateri uporabnik je to storil, uporabniško ime in njegov ID. Zapiše se kakšna operacija je potekala in v kateri tabeli podatkovne baze ter v kateri vrstici tabele. V primeru vstavljanja novih podatkov, se zapiše tudi kateri podatki so se vstavili. Pri tem se podatek, kot je recimo geslo, pri vstavljanju novega uporabnika izpusti.

2.8 Tipi uporabnikov

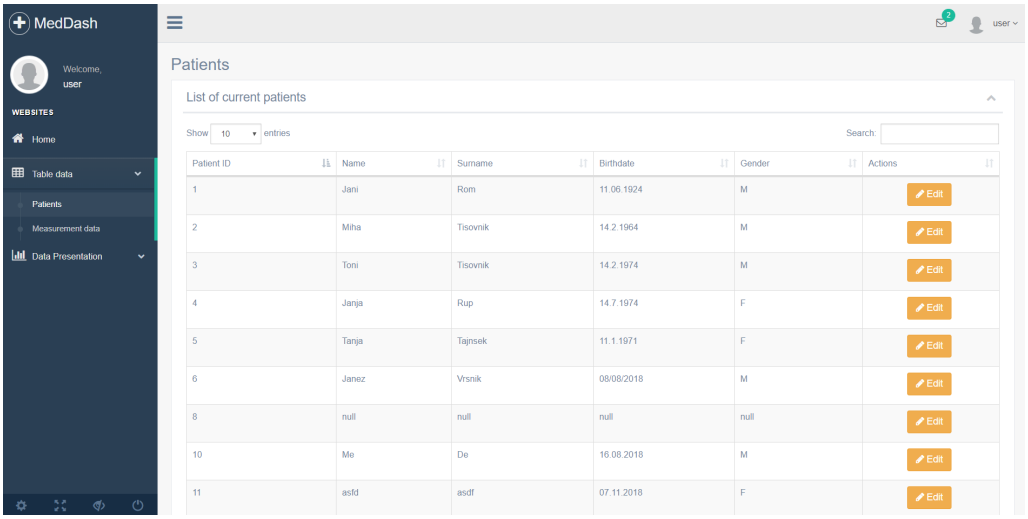
Aplikacija ima tri različne vrste uporabnikov. Dve vrsti sta pri tem namenjeni novim uporabnikov. Prva pa je narejena predvsem zaradi varnosti in je namenjena skrbiku aplikacije. Te tri vloge se imenujejo super administrator, administrator in uporabnik. Pri tem ima vsaka vloga različna dovoljenja in značilnosti.
Uporabnik lahko:

- vidi podatke o pacientih,
- vidi podatke o pacientovih meritvah,
- vidi grafične predstavitve pacientovih meritev,
- ureja podatke o pacientih.

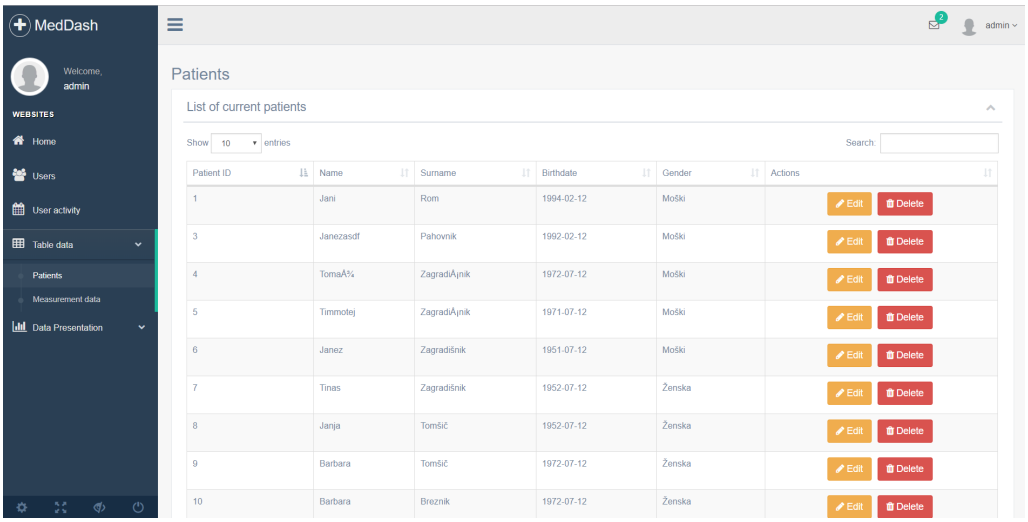
Administrator lahko:

- dodaja in spreminja ostale uporabnike,
- briše podatke o pacientih,
- vidi aktivnost ostalih uporabnikov.

Vloga super administratorja je dodana zaradi dodatne varnosti. Pri tem je edina razlika ta, da uporabnik superadmin ni viden vsem ostalim uporabnikov nikjer v aplikaciji, čeprav je podatek o njem shranjen v bazi. S tem lahko preprečimo, da bi eden od administratorjev razlastil vse ostale administrativne račune ali pa jih celo zbrisal in s tem pridobil nadzor nad aplikacijo.



Slika 2.31: Primer pogleda aplkacije, ki ga ima navaden uporabnik.



Slika 2.32: Primer pogleda aplkacije, ki ga ima administrator.

Administratorji lahko poleg dodajanja in odstranjevanja računa pregledajo tudi aktivnosti ostalih uporabnikov, ki so jih le ti izvajali v bazi. Okno samo je narejeno po modelu uporabnikovega profila, vendar vsebuje manj funkcionalosti, saj v njem lahko vidimo le osnovne podatke o uporabnikovem profilu in njegove aktivnosti.

Slika 2.33: Pogled za ogled aktivnosti drugega uporabnika.

Poglavje 3

Testiranje

Odločil sem se, da bom aplikacijo testiral ob vsaki večji spremembi. Na takšen način sem lahko hitreje odkril napake in jih popravil ter si s tem prihranil kasnejše delo.

V prvem koraku sem testiral podatkovno bazo in vrsto povezav, ki potekajo med njo in aplikacijot ter aplikacijo in uporabnikom. V tej stopnji razvoja aplikacije sem imel narejen le osnovi uporabniški vmesnik, ki je bil sposoben izvedbe CRUD operacij nad podatkovno bazo. Preveril sem ali se podatkovna baza ob izvedbi operacij pravilno posodablja in če so podatki, ki se shranijo v bazo pravilni. Preveril sem tudi ali prenos podatkov poteka po SSL povezavi.

Naslednji korak v testiranju je bilo preverjanje pravilnega delovanja avtentikacije in dostopa do virov. Pri tem koraku sem preveril možnost vpisa uporabnika s pravilnimi in napačnimi podatki ter ali lahko uporabnik dostopi do virov znotraj aplikacije brez da se je zgodil vpis. Zaradi določenih administrativnih funkcij znotraj aplikacije sem preveril, ali lahko uporabnik v primeru, da ni administrator, dostopi do le teh.

Za testiranjem osnovnega delovanja aplikacije sem se posvetil testiranju akcij, ki jih lahko izvede uporabnik in njihovem pravilnem obnašanju. Pri tem koraku sem preveril obnašanje povezav in gumbov nad katerimi uporabnik lahko izvaja akcije. Predvsem pa sem se posvetil vnosnim poljem. Poskušal

sem vnesti napačne informacije, nato sem opazoval kako se aplikacija na to odzove. Preveril sem tudi, če se slednje vnešene informacije pravilno posodobijo.

Zadnji korak, ki mi je še ostal je bil pregled uporabniškega vmesnika. V tem koraku sem predvsem testiral izgled aplikacije ob različnih ločljivost zaslona in kako se aplikacija obnaša ob njeni spremembi [4].

Poglavje 4

Zaključek

V diplomski nalogi je bila narejena spletna nadzorna plošča, ki teče na aplikacijskem strežniku. Aplikacija je namenjena podpori zdravnikom, ki bodo z njo lažje spremljali stanje bolnikov s parkinsonovo boleznijo. Aplikacija zagotavlja varnost podatkov med njihovim prenosom tako na povezavi strežnik podatkovna baza, kot na povezavi strežnik klient. To je potrebno zaradi medicinske narave podatkov. Aplikacija dinamično jemlje podatke iz baze in jih prikazuje na uporabniškem vmesniku. Tako lahko zdravnik pregleda podatke o pacientih in njihovih meritvah. Le te si lahko ogleda v tabeli ali grafični obliki.

Aplikacija zdravniku zagotavlja spremljanje meritev pacientov in vizualizacijo le teh. Tako zdravniku olajša spremljanje bolnikovega stanja skozi čas. Potencialno tudi omogoči razvoj umetne inteligence, ki bo iz podatkov pridobila koristne informacije. Aplikacija sama je namenjena uporabi s strani zdravnikov v ambulatnem okolju. Z njo bi lahko zdravnik med pregledom preučil trenutno stanje bolnika, kar bi mu olajšalo delo. Med pregledom podatkov, bi lahko tudi odkril, da se je stanje določenega bolnika zelo poslabšalo oziroma izboljšalo ter prilagodil njegov naslednji pregled glede na to informacijo.

V aplikaciji je tudi prostor za morebitne izboljšave. Uvedba umetna inteligence, ki bi lahko sama spremljala stanje bolnikov in bila sposobna oceniti

njihove meritve, bi aplikacijo precej izboljšala. S tem bi lahko zdravniku že aplikacija sama sporočila ali so meritve začele izstopati in kako se je stanje bolnika spremenilo. Zdravnik bi prihranil ogromno časa, saj bi lahko pregledal le podatke, ki so potencialno koristni.

Ostale potencialne izboljšave bi se nahajale predvsem pri obdelavi in prikazu podatkov. Zdravniku bi lahko prikazali več uporabnih statističnih podatkov o določenih meritvah, ki so potencialno koristni za spremljanje neregularnosti, tako pri meritvah samih, kot v stanju pacienta. S tem bi zdravniku omogočili tudi lažjo izločitev meritev, ki so se mogoče ponesrečile, naprimer pacientu pade telefon iz roke.

Pri grafični predstavitvi bi lahko razvili več različnih prikazov, ki bi zdravniku omogočali različen pogled na iste podatke. Tako bi lahko zdravnik pridobil več informacij iz danih podatkov, saj bi mu različni prikazi omogočili različne poglede nad trenutnim stanjem pacienta. Koristna bi bila tudi možnost primerjanja določenih podatkov, naprimer med mertivami samimi, določenimi obdobji ali med pacienti.

Literatura

[1] HTTP 403. Dosegljivo: https://en.wikipedia.org/wiki/HTTP_403, 2018. [Dostopano: 25. 8. 2018].

[2] Jonathan Anderson, John McRee, and Robb Wilson. *Effective UI: The Art of Building Great User Experience in Software*. O'Reilly Media, 2010.

[3] 10 nasvetov za boljše dashboarde. Dosegljivo: https://www.klipfolio.com/blog/10-tips-for-better-dashboards, 2013. [Dostopano: 5. 8. 2018].

[4] A.R. Fasolino and G.A. Di Lucca. Web application testing. Springer, 2006.

[5] JavaServer Pages. Dosegljivo: https://en.wikipedia.org/wiki/JavaServer_Pages, 2018. [Dostopano: 19. 8. 2018].

[6] James Martin. *Rapid application development*. Macmillan Publishing, 1991.

[7] Parkinsonova bolezen: Tremor. Dosegljivo: https://www.apdaparkinson.org/what-is-parkinsons/symptoms/tremor/, 2016. [Dostopano: 15. 8. 2018].